Learning Dynamical Models Using Motifs

Gregory Provan

Department of Computer Science, University College Cork, Ireland
{g.provan}@cs.ucc.ie

Abstract. Automatically creating dynamical system models, M, from data is an active research area for a range of real-world applications, such as systems biology and engineering. However, the overall inference complexity increases exponentially in terms of the number of variables in M. We solve this exponential growth by using canonical representations of system motifs (building blocks) to constrain the model search during automated model generation. The motifs provide a good prior set of building blocks from which we can generate system-level models, and the canonical representation provides a theoretically sound framework for modifying the equations to improve the initial models. We present an automated method for learning dynamical models from motifs, such that the models optimize a domain-specific performance metric. We demonstrate our approach on hydraulic systems models.

1 Introduction

Generating dynamical systems models remains a challenge for most real-world applications. For example, in systems biology, scientists aim to define genetic networks, using data acquired from high-throughput microarray analysis to identify the dynamical behaviour of gene clusters [2]. In automotive applications, engineers aim to develop control algorithms to guarantee driving performance and safety [4].

In these applications, researchers must develop for the dynamical model (1) the underlying structure (e.g., the equation form representing the structure of the gene cluster) and (2) the model's parameters. Among others, Voit [16] notes that structure identification is more difficult than parameter estimation.

Estimating dynamical systems models from data is computationally intensive. As the number of system states increases, the identification of highly coupled non-linear systems becomes increasingly challenging. In computational terms, the overall inference complexity for estimation increases exponentially in terms of the number of variables and parameters in a system [20].

Because computational blowups typically restrict the size of the learned dynamical system, most approaches attempt to restrict the search space. For example, in systems biology researchers use canonical versions of ordinary differential equations (ODEs) to transform learning arbitrary ODEs into learning parameters that characterize the ODEs [16]; in vision, some researchers have used human prior knowledge to learn partial differential equations (PDEs) describing the evolution of visual saliency [9].

Model libraries are widely used in systems design and engineering, and are the standard method for *hand-generating* large, complex models for several applications,

ranging from HVAC [19], fluid-flow [1], power [6] to biological [11, 15] systems. Motifs [12, 18], the building blocks of libraries for complex systems, are network sub-structures (i.e., frequently-occurring and unique sub-structures) from which large systems are built. Motifs can reduce design costs and speed up product development. However, a critical drawback is that the motif libraries are typically developed for simulation and design purposes, making them significantly less useful for other tasks such as control or diagnostics. Some tasks, e.g., control, require trading off model fidelity for computational efficiency; other tasks require additional information, e.g., diagnostics requires simulation behaviours for both faulty and nominal conditions.

Because design-focused motif libraries are not optimized for non-design tasks, companies typically create application-specific system-level models that do not leverage existing motif libraries. To avoid this duplication of effort, we propose an approach for learning task-specific system-level models using motif libraries: given data $\mathcal{D}$ concerning the selected task $\mathcal{T}$, we search over the space of motif configurations and parameters to generate a system model that optimises a metric μ over $\mathcal{D}$. Our work differs from prior work that learns dynamical equations purely from data, e.g., [13, 17], in that we generate models from a pre-defined motif library, measuring the models' performance using statistical model-comparison techniques [10].

We solve this exponential growth problem by constraining the model search during automated model generation, using canonical representations of pre-defined motifs. The canonical representation significantly constrains the structure and parameter space, and the motif libraries provide a good prior set of building blocks from which to generate system-level models.

In this article we show how to reduce the enormous search space of model and parameter configurations by using motif-based learning. We learn task-specific motifs by (1) imposing physical constraints over the models, and (2) using a canonical representation for non-linear dynamical equations [14] that allows us to define a clear notion of relative model fidelity, which we use for guiding search.

Our contributions are as follows.

- We develop a task-specific method for generating system models that can trade off model fidelity for model size, so that we can tailor models for tasks requiring smaller (more efficient) models.
- We show how motifs can improve the computational efficiency of learning task-specific systems that are modelled using non-linear dynamical equations by at least one order-of-magnitude.
- We illustrate our approach on a non-linear tank benchmark.

2 Notation

2.1 Model Representation

We represent a system as an inter-connected set of sub-systems (or components), using notions standard within the computing and engineering literature, e.g., [3]. We assume that for any system, we have the system topology, which specifies the component-type

connections. We further assume that we have a library of motifs (or components), where we specify each motif using multiple levels of fidelity.

We characterize a system/motif using a graph $G(V, E)$ of vertices and edges, and a set $\mathcal{E}_G$ of equations. The graph specifies the relationships among the variables in $\mathcal{E}_G$ and the equations specify the transformations occurring in the system/motif. For example, a bio-system might define converting 2 units of species A into 1 unit of B: $\overset{2}{A} \rightleftharpoons \overset{1}{B}$. We can represent this using a graph consisting of vertex A connected to vertex B, with a differential equation denoting the rate of conversion of A to B using parameters θ.

As shown in Figure 2(b), a motif consists of sets of input and output variables, $\boldsymbol{v}_i$ and $\boldsymbol{v}_o$ respectively, a set $\boldsymbol{\theta}$ of parameters, and a set $\mathcal{E}$ of equations over the variables $\boldsymbol{v}$ and parameters $\boldsymbol{\theta}$.

A sub-system/component can consist of a hierarchical composition of sub-components. A *base-level component* includes no embedded sub-components.

In most applications there is uncertainty concerning the structure and equational form of a motif, i.e., the level of fidelity or detail of the structure/equations. We capture that uncertainty by specifying mutiple versions of any motif, denoted $\lambda = \{\lambda_1, ..., \lambda_k\}$. we can also define a distribution over λ to denote the relative likelihood of each version being correct.

In this article, we adopt the Modelica component-based representation [3]. A motif corresponds to a Modelica component model, which uses the following three entities to define a system:

1. Component library;
2. System topology (represented as a graph G);
3. A connection mechanism (with clear semantics).

Component library: The component library consists of a collection of components of different types, e.g., resistor, capacitor, inductor, etc.

System topology: We assume that a system has an associated topology $G(C, E)$, where each node $\chi \in C$ in G defines a component χ and each edge $e = (\chi_i, \chi_j) \in E$ defines a connection between χ_i and χ_j.

Connection mechanism: Components are connected via the connection mechanism, according to the system topology G. In Modelica, connectors define the variables for the component communication interface, i.e., connectors are instances of connector classes. Connectors specify external interfaces for interaction between components.

The component framework specifies a set of constraints over components and connections. For example, we constrain a connection to be possible only between connectors of equivalent type. We define, WLOG, input (positive) and output (negative) connectors.

2.2 Dynamical Systems

We consider dynamical systems that can be described by a set of (noise-free) ODEs:

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \psi(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\theta}), \boldsymbol{x}(t_0) = x_0, \\ \boldsymbol{y}(t) &= \varphi(\boldsymbol{x}(t), \boldsymbol{\theta}), \end{aligned} \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u}(t) \in \mathbb{R}^q$ is the input vector, $\boldsymbol{y}(t) \in \mathbb{R}^m$ is the observation vector, and $\boldsymbol{\theta} \in \mathbb{R}^p$ is a vector of parameters associated with the state transition and control dynamics, and the observation function. $\phi_{\boldsymbol{x}_0}(\boldsymbol{\theta}, \boldsymbol{u})$ denotes the input-output mapping of the system (1) started at the initial state $\boldsymbol{x}_0$ with parameter set θ. We use $\boldsymbol{v} = \{\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u}\}$ to denote the set of system variables.

2.3 Thermo-Hydraulic Example: Tank System

Throughout this article we will use as a running example a thermo-hydraulic system: a system of interconnected tanks. In this domain, the motifs consist of tanks, valves, pipes, pumps, flow sources/sinks, etc. Systems comprising these motifs are used to model several domains, including hydrology (tanks and pipes correspond to lakes and rivers, respectively), chemical process control, and cardiovascular systems.

In this article we consider three components, tanks, valves, and pipes. Figure 1(a) shows the three components. For each component we define a block diagram with a

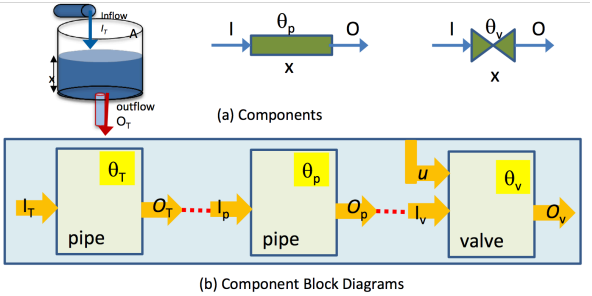

Fig. 1. System components: tank, pipe and valve, and their block diagrams

single input and output, denoting inflow and outflow respectively.

We now describe in more detail the model of a single tank, which is an example of a basic component model. Figure 2(a) shows the tank component. We model a tank component using a set of equations over variables denoting inputs (fluid pressure p_{in}, and fluid flow q_{in}) and outputs (p_{out}, q_{out}). The tank has cross-sectional area of A, and outlet cross-section a. If the height h of fluid in the tank changes at a rate $\dot{h}$, then we have $A_1\dot{h} = q_{in} - q_{out}$. We assume that we measure only pressure, using $p = gh$, where g is a gravity parameter. We could also define equations using parameters like viscosity σ. Taken together, the tank thus comprises a motif/component with variables, parameters and equations as shown in Figure 2(c).

We define a tank sub-system to consist of three connected motifs: a tank, a valve, and a pipe, as shown in Figure 1(b). We can control the flow through this system, and the height of fluid in the tank, by controlling the inflow to the tank and the valve setting.

We can extend our example by defining the three-tank system shown in Fig. 3, which consists of three connected tank sub-systems. Tank T_i has area A_i and inflow q_{i-1}, for

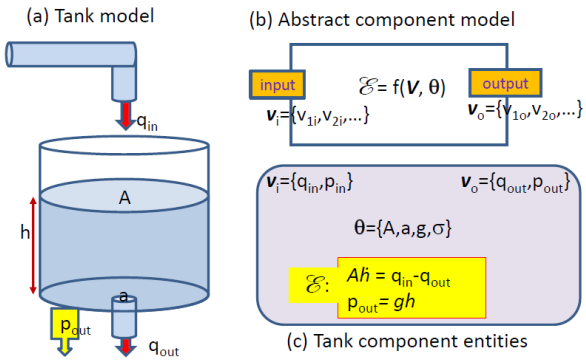

Fig. 2. Model for tank component

$i = 1, 2, 3$. We create this system by connecting together three tank components, with a valve V_i regulating the flow q_i out of tank i, for $i = 1, 2, 3$.

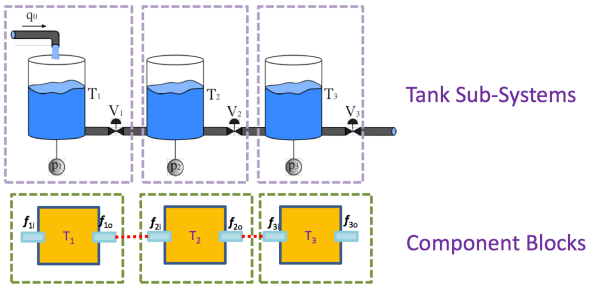

Fig. 3. Diagram of the three-tank system.

Tank T_1 gets filled from a pipe, with measured flow q_0. Hence, our control input is $u = \{q_0\}$.

We assume that we don't directly measure any flows other than the inlet flow q_0. Therefore, we use the tank heights as a proxy for deriving flows through the multi-tank system.

Torricelli's Law defines the flow q_i out of tank i, with liquid level h_i, into tank j, as:

$$q_i = \gamma sign(h_i - h_j)\sqrt{2g(h_i - h_j)}, \tag{2}$$

where the coefficient γ models the area of the drainage hole and its friction factor through the hole.

We can use equation 3 to derive the following equations for the three-tank system shown in Fig. 3:

$$\begin{aligned}\dot{h}_1 &= q_0 - c_1\sqrt{h_1 - h_2} \\ \dot{h}_2 &= c_2\sqrt{h_1 - h_2} - c_3\sqrt{h_2 - h_3}, \\ \dot{h}_3 &= c_4\sqrt{h_2 - h_3} - c_5\sqrt{h_3},\end{aligned} \tag{3}$$

where the constants $c_1, \cdots, c_5$ summarize the system parameters representing cross-sectional areas, friction factors, gravity, etc. We measure tank pressure, whose equations are given by $p_i = h_i g$, for $i = 1, 2, 3$. Consequently, we define the parameter set using $\Theta = \{c_1, c_2, c_3, c_4, c_5, g\}$.

3 Learning System Models

3.1 Objective

This section describes our approach for learning system models. Our objective is to search over the space $\mathcal{S}$ of possible models (composed from component models of different fidelity) to identify a model that optimizes a criterion μ:

$$M^* = \underset{i:M_i \in \mathcal{S}}{\operatorname{argmax}}\, \mu(M_i|\mathcal{D}) \tag{4}$$

We use a statistical approach for this model search, first assigning a prior distribution $P(\mathcal{S})$ over the model space. We then use the structure defined by the canonical models to provide us with a framework for model search. Finally, we must use a stopping criterion to identify when we have achieved an "optimal" model given $(\mu, \mathcal{D})$.

Our experiments test the hypothesis that our proposed approach is computationally more efficient than and loses little simulation accuracy relative to the standard (unconstrained) approach.

3.2 System Architecture

Figure 4 shows the system architecture for model learning.
We adopt a two-step process for learning a target model M. We first learn the the motif/component library $\mathcal{L}$; we then use this to constrain the induction of the full system topology G. We represent each component as a set $\mathcal{E}$ of dynamical equations.

Motif Learning We use various system constraints to learn a motif/component library. We map the input equations $\mathcal{E}$ into a canonical representation, defined over $\langle \boldsymbol{z}, \boldsymbol{\pi} \rangle$, where $\boldsymbol{z}$ is a set of transformed variables and $\boldsymbol{\pi}$ is the set of canonical parameters.

System-Level Model Learning We search over $\tilde{\boldsymbol{\pi}}$ to select the model that optimizes metric μ. We use a statistical model-comparison tool [10] to determine the best model. We ensure a computationally tractable search space by mapping $\langle \boldsymbol{z}, \boldsymbol{\pi} \rangle$ to a subset $\tilde{\boldsymbol{\pi}}$ of the full model parameter space using physical constraints.

We can search over the model space in two ways:

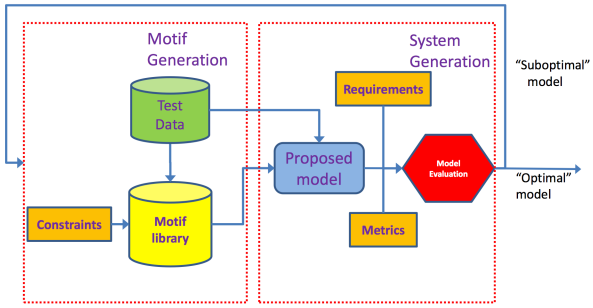

Fig. 4. System architecture for learning models, based on (a) learning motifs and (b) using the motifs to constrain learning complete models.

Unconstrained Without motifs, our learning task must define the structure and parameters of the component model equations during the system-level learning process. Given the variability of defining ODEs, we use a gradient-descent search that systematically modifies the set W of ODEs to achieve a targeted change in the simulation performance of W.

Motif-Based Here, we assume that we use the motif/component models to constrain system-level search. We search over all component combinations of a set of models of different fidelity to generate a system model optimizing our metric μ. If we have k models for every component, and a system consists of l components, then the space of models to be searched increases exponentially with the number of components (k^l), i.e., the search space becomes prohibitively large for large systems ($l > 100$).

3.3 Tank System Canonical Representation

This section describes how we define a canonical representation for our tank model. We first define the representation, and then use our tank model as an example. We adopt a Power-Law Canonical Representation, also called S-systems [14], for constraining model search. This representation is general, since almost all non-linear models can be exactly recast into power-law models through a transformation using auxiliary variables, as specified in Theorem 1. This mapping generates $m - n$ additional constraints beyond the n original equations: see [14] for details.

Theorem 1 ([14]). *Let*

$$\dot{x}_i = f_i(x_1, x_2, ..., x_n), \quad x_i(0) = x_{i0}, \quad i = 1, 2, ..., n \tag{5}$$

be a set of differential equations where each f_i consists of sums and products of elementary functions, or nested elementary functions of elementary functions. Then there is a smooth change of variables $\boldsymbol{x} \rightarrow \boldsymbol{z}$ that recasts Equations 6 into a power-law (or

S-) system $\langle \boldsymbol{z} = \{z_1, ..., z_{m+n}\}, \boldsymbol{\pi} \rangle$*:*

$$\dot{z}_i = \alpha_i \prod_{j=1}^{n} z_j^{\gamma_{ij}} - \beta_i \prod_{j=1}^{n} z_j^{\zeta_{ij}}, \quad z_i(0) = z_{i0}, \;\; i = 1, 2, ..., m \tag{6}$$

where z_i *are real non-negative variables, and the parameters* $\boldsymbol{\pi} = \{\alpha_i,\ \beta_i, \gamma_{ij},\ \zeta_{ij}\}$ *are such that* $\alpha_i,\ \beta_i$ *are real non-negative and* $\gamma_{ij},\ \zeta_{ij}$ *are real.*

In the following we will show how this representation, together enforced physical constraints, can significantly reduce the search space for learning dynamical systems, through a principled control of the power-law system variables and parameters.

We now show how we can transform the tank model into a power-law model. We transform the state equations (Eq. 4) for this 3-tank system to a power-law model by making the following substitutions:

$$\begin{aligned} x_0 &\leftarrow q_0 \\ x_1 &\leftarrow h_1 \\ x_2 &\leftarrow h_2 \\ x_3 &\leftarrow h_3 \\ x_4 &\leftarrow h_1 - h_2 \\ x_5 &\leftarrow h_2 - h_3 \end{aligned}$$

This can be expressed in the general formula as follows:

$$\begin{aligned} \dot{x}_1 &= x_0 - \beta_1 x_4^{\zeta_{11}} \\ \dot{x}_2 &= \alpha_2 x_4^{\gamma_{21}} - \beta_2 x_5^{\zeta_{21}} \\ \dot{x}_3 &= \alpha_3 x_5^{\gamma_{31}} - \beta_3 x_3^{\zeta_{31}} \\ \dot{x}_4 &= (\alpha_{41} x_5^{\gamma_{41}} + x_0) - \beta_4 x_4^{\zeta_{41}} \\ \dot{x}_5 &= \alpha_5 x_6^{\gamma_{51}} - \beta_5 x_5^{\zeta_{51}} \\ \dot{x}_6 &= \alpha_6 x_4^{\gamma_{61}} - \beta_6 x_5^{\zeta_{61}} \end{aligned} \tag{7}$$

where most γ_{ij} and ζ_{ij} are $\frac{1}{2}$. We assume that $\boldsymbol{x}(0) = (0, 0, 0, 0, 0, 0)$.

In this transformation, the equations for $\dot{x}_1,\ \dot{x}_2,\ \dot{x}_3$ comprise the transformed state equations, and the latter 3 equations are constraints. We can see that each state equation (corresponding to a tank equation) specifies an (inflow - outflow) representation.

3.4 Constraints on the Model Search Space

We must search over a vast space of models if no constraints are imposed. If we fix the variable specification ($\boldsymbol{x}$, denoting the x_i's) in equation 1, a search over the space of models must consider the entire parameter space, in the worst case. If we modify the variable specification ($\boldsymbol{x}$), then we must consider the parameter space for every setting of variables $\boldsymbol{x}$. In a power-law model, for each component/system, the parameters are $\boldsymbol{\theta} = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}, \boldsymbol{\zeta}\}$, where $\alpha_i,\ \beta_i$ are real non-negative, $\gamma_{ij},\ \zeta_{ij}$ are real.

We can significantly reduce the search space by using well-known physical constraints, transforming this space from a multi-dimensional continuous-valued space to a finite discrete-valued space. As an example, consider the tank system: component i, $i = 1, 2., , ,$ has 4 parameters in the canonical model, $\boldsymbol{\theta} = \{\alpha_i, \beta_i, \boldsymbol{\gamma}, \boldsymbol{\zeta}\}$, of which two (α_i, β_i) are multiplicative parameters, and two $(\boldsymbol{\gamma}, \boldsymbol{\zeta})$ are exponents for variables.

Given an assignment of $(\boldsymbol{\gamma}, \boldsymbol{\zeta})$, the estimation of the multiplicative parameters given data $\mathcal{D}$ is a well-known, highly-studied process, e.g., [21, 8]. It is the exponent-based parameters $(\boldsymbol{\gamma}, \boldsymbol{\zeta})$ that create a difficult search problem, a problem that has received relatively little attention. Without any constraints, we must search over the real-valued space of $\boldsymbol{\gamma} \times \boldsymbol{\zeta}$. However, we use physical (or model-based) constraints to prune the search space significantly. Consider the tank example: the physics-based model is non-linear, and corresponds to setting all γ_{ij} and ζ_{ij} to be $\frac{1}{2}$ for $i = 1, ...5$ and to $-\frac{1}{2}$ for $i = 6$. Other versions of this model that are typically analysed are *constant*, obtained by setting all γ_{ij} and ζ_{ij} to be 0, or *linear*, obtained by setting all γ_{ij} and ζ_{ij} to be 1:

$$
\begin{aligned}
\dot{x}_1 &= x_0 - \beta_1 x_4 \\
\dot{x}_2 &= \alpha_2 x_4 - \beta_2 x_5 \\
\dot{x}_3 &= \alpha_3 x_5 - \beta_3 x_3 \\
\dot{x}_4 &= (\alpha_{41} x_5 + x_0) - \beta_4 x_4 \\
\dot{x}_5 &= \alpha_5 x_6 - \beta_5 x_5 \\
\dot{x}_6 &= \alpha_6 x_4 - \beta_6 x_5
\end{aligned}
$$

If we want to search over a tank component, this is equivalent to searching over the space described by the equation for $\dot{x}_2$: $\alpha_2 x_4^{\gamma_{21}} - \beta_2 x_5^{\zeta_{21}}$. The three most typical classes of model, constant, non-linear and linear, correspond to setting each of (γ, ζ) to $\{0, \frac{1}{2}, 1\}$, i.e., 0 corresponds to constant, $\frac{1}{2}$ to non-linear, and 1 to linear. Table 1 shows these different models.

γ_{21}	ζ_{21}	$\dot{x}_2$	Original	Type
0	0	$\alpha_2 - \beta_2$	$\alpha_2 - \beta_2$	constant
1	1	$\alpha_2 x_4 - \beta_2 x_5$	$\alpha_2(h_1 - h_2) - \beta_2(h_2 - h_3)$	linear
$\frac{1}{2}$	$\frac{1}{2}$	$\alpha_2 x_4^{\frac{1}{2}} - \beta_2 x_5^{\frac{1}{2}}$	$\alpha_2(h_1 - h_2)^{\frac{1}{2}} - \beta_2(h_2 - h_3)^{\frac{1}{2}}$	non-linear

Table 1. Hierarchy of equations for tank 2

We can create a hierarchy by fixing (α, β) and varying (γ, ζ), as shown in Table 1. In this table, we show the plausible values of these parameters, namely $\{0, \frac{1}{2}, 1\}$, corresponding to constant, non-linear and linear equations. We could, in theory, examine model combinations with all discrete combinations of (α, β), or the infinite combinations of real-valued pairs, if (α, β) are both reals. However, by enforcing a restriction to constant, non-linear and linear equations, we obtain a lattice-structured search space with a relatively small, finite number of (α, β)-combinations to search.

The lattice defines a clear notion of relative model fidelity. A directed edge from model A to model B means that model B has one component whose equations are

more complex (and probably lead to higher-fidelity inference) than those in model A. Traversing the lattice thus entails traversing the model space, exploring models with well-defined differences in relative model fidelity.

4 Empirical Analysis

This section describes our empirical analysis of the proposed power-law framework.

4.1 Experiments: Motif-Constrained Search

We have run experiments on the 3-tank system, using a collection of models for tanks and valves, namely constant, linear and non-linear instances of each, giving a search space of 729 possible models, each with a very large parameter space. We used the fully non-linear model as the gold standard model M^*, with parameters specified in [7], to simulate data that we used for learning. We computed the sum-of-squared-error (SSE) difference between M^* and learned model. We tested both breadth-first and depth-first search algorithms in the parameter lattice, starting from the constant model as the root of the search tree. The results indicate that the depth-first search algorithm is more efficient.

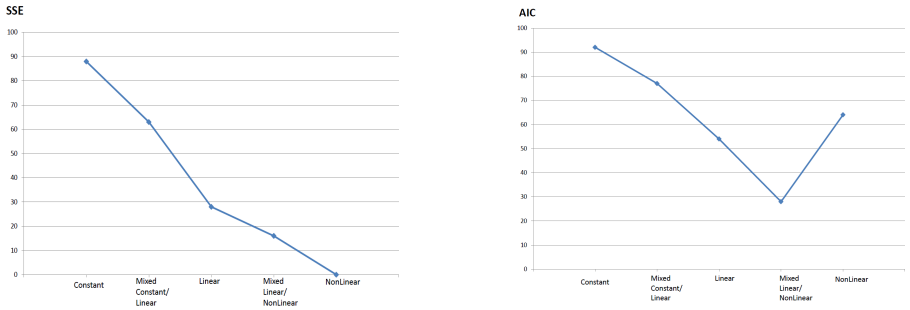

Fig. 5. Comparison of SSE and AIC scores for composed models

However, when we penalize a model for its number of parameters in addition to penalizing a relative lack of accuracy, as in the AIC metric [5], the mixed linear/nonlinear model scores best. Figure 5 compares the AIC scores for composed models of the 3-tank system.

4.2 Experiments: Unconstrained

We ran experiments in which we modified the canonical equation structure in a continuous manner. We can simplify the the initial set of equations (8) by setting various γ and ζ parameters to zero, or we can extend the equations by increasing the order of

various γ and ζ parameters or by adding multiplicative x_j variables into the equations. The benefits of the canonical equation structure is that it constrains how to modify the equations, and it provides a clear framework to simplify or extend a set of equations.

Figure 6 compares the AIC scores for composed models of the 3-tank system, when we generate models by model extension. Here, we also create higher-order nonlinear models, which are increasingly penalized by the AIC metric due to their increasing number of parameters outweighing the improved model simulation accuracy.

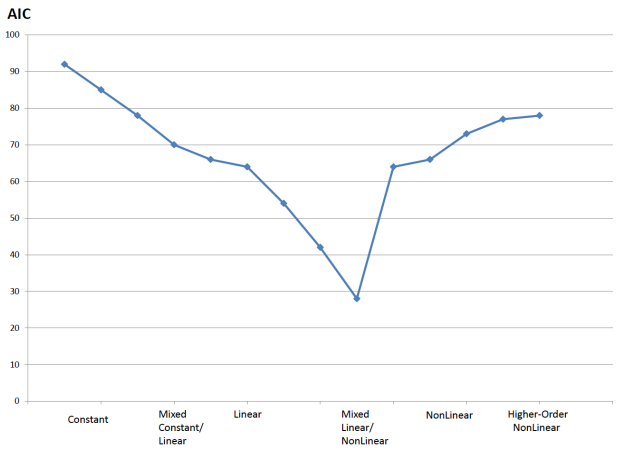

Fig. 6. Comparison of AIC scores for extended models

4.3 Discussion

Our results show the tradeoffs we can study by automatically generating models. Although the simulation accuracy increases for the nonlinear models over simpler models, then parameter estimation may possibly be too costly. The AIC metric provides a measure that addresses this trade-off.

A second outcome is that the model extension approach is significantly more expensive computationally than the composition approach. In the hydraulic domain studied model extension did not create significantly more accurate models than those composed from the multi-fidelity library, although this is probably domain-dependent.

5 Conclusions

This article has illustrated a system that uses a library of motifs to create a computationally efficient system for learning task-specific ODE models from data. In particular, our approach can trade off model fidelity for model size (which typically corresponds to inference complexity) on a task by task basis.

Acknowledgement. This research was supported by SFI grants 12/RC/2289 and 13/RC/2094.

References

1. Behbahani-Nejad, M., Bagheri, A.: The accuracy and efficiency of a matlab-simulink library for transient flow simulation of gas pipelines and networks. Journal of Petroleum Science and Engineering 70(3), 256–265 (2010)
2. Busetto, A.G., Hauser, A., Krummenacher, G., Sunnåker, M., Dimopoulos, S., Ong, C.S., Stelling, J., Buhmann, J.M.: Near-optimal experimental design for model selection in systems biology. Bioinformatics 29(20), 2625–2632 (2013)
3. Fritzson, P.: Principles of object-oriented modeling and simulation with Modelica 2.1. John Wiley & Sons (2010)
4. Gao, D.W., Mi, C., Emadi, A.: Modeling and simulation of electric and hybrid vehicles. Proceedings of the IEEE 95(4), 729–745 (2007)
5. Hu, S.: Akaike information criterion. Center for Research in Scientific Computation (2007)
6. Jan, J., Šulc, B.: User friendly simulink thermal power plant modelling using object oriented non-linear dynamic model library. In: 5th IASTED International Conference Power and Energy Systems (PES 2001) Tampa. pp. 253–256 (2001)
7. Join, C., Sira-Ramírez, H., Fliess, M.: Control of an uncertain three-tank system via on-line parameter identification and fault detection. In: IFAC World Congress (July 2005)
8. Kravaris, C., Hahn, J., Chu, Y.: Advances and selected recent developments in state and parameter estimation. Computers & Chemical Engineering 51, 111–123 (2013)
9. Liu, R., Cao, J., Lin, Z., Shan, S.: Adaptive partial differential equation learning for visual saliency detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3866–3873 (2014)
10. Marin, J.M., Pudlo, P., Robert, C.P., Ryder, R.J.: Approximate bayesian computational methods. Statistics and Computing 22(6), 1167–1180 (2012)
11. Mateják, M., Kulhánek, T., Šilar, J., Privitzer, P., Ježek, F., Kofránek, J.: Physiolibrary: Modelica library for Physiology. In: 10th International Modelica Conference. pp. 499–505. Linköping University Electronic Press Lund, Sweden (2014)
12. Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., Alon, U.: Network motifs: simple building blocks of complex networks. Science 298(5594), 824–827 (2002)
13. North, B., Blake, A.: Learning dynamical models using expectation-maximisation. In: Computer Vision, 1998. Sixth International Conference on. pp. 384–389. IEEE (1998)
14. Savageau, M.A., Voit, E.O.: Recasting nonlinear differential equations as S-systems: a canonical nonlinear form. Mathematical biosciences 87(1), 83–115 (1987)
15. Seo, S.W., Jung, G.Y.: Synthetic regulatory RNAs as tools for engineering biological systems: Design and applications. Chemical Engineering Science 103, 36–41 (2013)
16. Voit, E.O.: Biochemical systems theory: a review. ISRN Biomathematics 2013 (2013)
17. Wahlström, N., Schön, T.B., Deisenroth, M.P.: Learning deep dynamical models from image pixels. IFAC-PapersOnLine 48(28), 1059–1064 (2015)
18. Weihe, K.: Motifs in networks. In: Gems of Combinatorial Optimization and Graph Algorithms, pp. 59–68. Springer (2015)
19. Wetter, M., Zuo, W., Nouidui, T.S., Pang, X.: Modelica buildings library. Journal of Building Performance Simulation 7(4), 253–270 (2014)
20. Wu, S.J., Wu, C.T.: Seeding-inspired chemotaxis genetic algorithm for the inference of biological systems. Computational biology and chemistry 53, 292–307 (2014)
21. Young, P.: Parameter estimation for Continuous-time Models: a Survey. Automatica 17(1), 23–39 (1981)